Kuka API : Manuel de référence

Généré par Doxygen 1.3.2

Mon Sep 8 09 :17 :19 2003

Table des matières

Chapitre 1

Kuka API.

```
0.0.5

KUKA API README
---------------

        kuka_api


------------------------------------------------------------------------------
 -> Pour toutes questions vous pouvez me contacter  chavent@imerir.com.
    Vos remarques et suggestions sont bienvenues. Merci.
------------------------------------------------------------------------------

------------------------------------------------------------------------------
        Description :
------------------------------------------------------------------------------

        * Ce projet vise  fournir une interface de programmation pour les
          robots Kuka.


------------------------------------------------------------------------------
        Utilisation
------------------------------------------------------------------------------
           __________
         _| BINAIRES |________________________________________________________
        |
        |1- Installation
        |  Si vous avez un packetage de la forme kuka_api-Version-Arch-1.tgz
        |  vous pouvez dcompresser le contenu de l'archive dans le dossier
        |  qui vous convient :
        |  $: tar -zxvf kuka_api-Version-Arch-1.tgz
        |  Sous Slackware vous pouvez faire :
        |  $: insallpkg kuka_api-Version-Arch-1.tgz
        |
        |2- Utilisation
        |  Vous pouvez lire les manuels qui sont dans share/doc :
        |       - kuka_api_man_ref.pdf pour le dtail de chaque fonctions...
        |       - kuka_api_man_util.pdf pour etre guid  travers un example.
        |

                      _________
         ____________| SOURCES |______________________________________________
        |
        |1- Linux
        |  La compilation des packetages fut faite sur Slackware 9.
        |  Pour compiler la librairie il faut : gcc, make, rpcgen, rpc.h,
```

```
|  libnsl.a.
|  Taper :
|  $: make src examples
|
|  Pour compiler le manuel de rfrence il faut doxygen.
|  Pour compiler le manuel d’utilisateur il faut latex.
|  $: make doc
|
|
|2- Windows
|  Il faut se place dans le rpertoire src/win et taper :
|  $: make
|  Ceci compile la librairie et c’est tout !
|  Les docs sont rcuprables en ligne au format pdf, et les
|  examples, vous les compilerez avec ce que vous voulez.
|


-----------------------------------------------------------------------------
        Notes
-----------------------------------------------------------------------------

        La version 0.0.4 du client est compatibles avec le serveur 0.0.2,
        0.0.3, et 0.0.4.

        Attention  partir de la version 0.0.4, l’interface du client change.
        Ce dernier reste compatible avec les serveurs antrieurs, mais les
        application programme pour fonctionner avec les version antrieures
        devrons etre modifies.
```

Chapitre 2

Kuka API : Index des composants

2.1 Kuka API : Liste des composants

Liste des classes, des structures et des unions avec une brève description :

Chapitre 3

Kuka API : Index des fichiers

3.1 Kuka API : Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Kuka API : Index des pages

4.1 Kuka API : Pages associées

Liste de toutes les pages de documentation associées :

Chapitre 5

Kuka API : Documentation de la classe

5.1 Référence de la structure kukaAxis_s

Description d'une configuration en fonction de chaque articulation.

```
#include <kuka_api.h>
```

Attributs Publics

- kukaReal_t a1
- kukaReal_t a2
- kukaReal_t a3
- kukaReal_t a4
- kukaReal_t a5
- kukaReal_t a6

5.1.1 Description détaillée

Description d'une configuration en fonction de chaque articulation.

Structure permettant de décrire une position spécifique aux axes. Ici, comme chaque axe est un pivot, les composantes de cette structure sont des angles.

À Faire

implémenter le type E6AXIS

5.1.2 Documentation des données imbriquées

5.1.2.1 kukaReal_t a1

Rotation autour de l'axe 1

5.1.2.2 kukaReal_t a2

rotation autour de l'axe 2

5.1.2.3 kukaReal_t a3

rotation autour de l'axe 3

5.1.2.4 kukaReal_t a4

rotation autour de l'axe 4

5.1.2.5 kukaReal_t a5

rotation autour de l'axe 5

5.1.2.6 kukaReal_t a6

rotation autour de l'axe 6

La documentation associée à cette structure a été générée à partir du fichier suivant :
– kuka_api.h

5.2 Référence de la structure kukaError_s

Description des erreurs.

```
#include <kuka_api.h>
```

Attributs Publics

- kukaErrorType_t type
- long no
- kukaString_t desc

5.2.1 Description détaillée

Description des erreurs.

Grace à cette structure les erreurs sont décrites en fonction de :
- leur provenance (leur type).
- leur numéro (qui pour l'instant sert surtout pour récupérer la description des erreurs de type KUKA_-KUKA_E).
- leur description.

5.2.2 Documentation des données imbriquées

5.2.2.1 kukaErrorType_t type

Le type d'erreur dépend de sa source. Si l'erreur est générée par :
- le robot (coté serveur donc) : type = KUKA_KUKA_E
- les appels de fonctions d'accés au composant com crosscommexe (coté serveur donc) : type = KUKA_-CROSS_E
- les appels de fonctions rpc (coté client donc) : type = KUKA_RPC_E
- les appels de fonctions de l'api (coté client donc) : type = KUKA_API_E

5.2.2.2 long no

Ce numéro sert, pour l'instant, surtout à récupérer la description des erreurs de type KUKA_KUKA_E.

5.2.2.3 kukaString_t desc

La description. Tant que possible elle a été faite de manière suivant : "source : description"

La documentation associée à cette structure a été générée à partir du fichier suivant :
- kuka_api.h

5.3 Référence de la structure kukaFrame_s

Description d'un point (position/orientation) dans l'espace.

`#include <kuka_api.h>`

Attributs Publics

- kukaReal_t x
- kukaReal_t y
- kukaReal_t z
- kukaReal_t a
- kukaReal_t b
- kukaReal_t c

5.3.1 Description détaillée

Description d'un point (position/orientation) dans l'espace.

Cette structure représente le type de donnée 'frame' du krl. Elle permet de représenter l'orientation et la position d'un point dans l'espace.

Les composantes a,b et c représentent le roulis, le tangage et l'embardée. Plus précisément :
- a est l'angle de rotation autour de Z,
- b est l'angle de rotation autour de Y,
- c est l'angle de rotation autour de X.

5.3.2 Documentation des données imbriquées

5.3.2.1 kukaReal_t x

Composante x des coordonnées cartésiennes du point.

5.3.2.2 kukaReal_t y

Composante y des coordonnées cartésiennes du point.

5.3.2.3 kukaReal_t z

Composante z des coordonnées cartésiennes du point.

5.3.2.4 kukaReal_t a

Angle de rotation autour de z

5.3.2.5 kukaReal_t b

Angle de rotation autour de y

5.3.2.6 kukaReal_t c

Angle de rotation autour de x

La documentation associée à cette structure a été générée à partir du fichier suivant :
– kuka_api.h

5.4 Référence de la structure kukaPos_s

Description sans equivoque d'un point (position/orientation).

```
#include <kuka_api.h>
```

Attributs Publics

- kukaReal_t x
- kukaReal_t y
- kukaReal_t z
- kukaReal_t a
- kukaReal_t b
- kukaReal_t c
- kukaInt_t s
- kukaInt_t t

5.4.1 Description détaillée

Description sans equivoque d'un point (position/orientation).

Cette structure représente le type de donnée 'pos' du krl. Bien que le type kukaFrame_t soit suffisant et sans equivoque pour décrire des coordonnées, il y a parfois plusieurs configuration d'axe possibles pour atteindre un point. Donc grace à s et t on lève l'ambiguité. Pour une description détaillée des membres x,y,z,a,b et c de la structure voir kukaFrame_s.

Voir galement :

kukaFrame_s

À Faire

implémenter le type E6POS

5.4.2 Documentation des données imbriquées

5.4.2.1 kukaReal_t x

5.4.2.2 kukaReal_t y

5.4.2.3 kukaReal_t z

5.4.2.4 kukaReal_t a

5.4.2.5 kukaReal_t b

5.4.2.6 kukaReal_t c

5.4.2.7 kukaInt_t s

5.4.2.8 kukaInt_t t

La documentation associée à cette structure a été générée à partir du fichier suivant :

- kuka_api.h

5.5 Référence de la structure kukaVal

Union de valeurs typées.

`#include <kuka_api.h>`

Attributs Publics

- kukaType_t type
- union {
  kukaInt_t kukaInt
  kukaReal_t kukaReal
  kukaBool_t kukaBool
  kukaChar_t kukaChar
  kukaString_t kukaString
  kukaAxis_t kukaAxis
  kukaFrame_t kukaFrame
  kukaPos_t kukaPos
  kukaError_t kukaError
  } kukaVal_u

5.5.1 Description détaillée

Union de valeurs typées.

Union des valeurs Si type=KUKA_XYZ alors kukaVal_u.kukaXyz est de type kukaXyz_t.

Par exemple si nous voulons une variable de type position nous aurons :

```
variable.nom="$POS_ACT";
variable.valeur.type=KUKA_POS;
variable.valeur.kukaVal_u.kukaPos.x=(kukaFloat_t)3.5;
```

5.5.2 Documentation des données imbriquées

5.5.2.1 kukaType_t type

5.5.2.2 kukaInt_t kukaInt

5.5.2.3 kukaReal_t kukaReal

5.5.2.4 kukaBool_t kukaBool

5.5.2.5 kukaChar_t kukaChar

5.5.2.6 kukaString_t kukaString

5.5.2.7 kukaAxis_t kukaAxis

5.5.2.8 kukaFrame_t kukaFrame

5.5.2.9 kukaPos_t kukaPos

5.5.2.10 kukaError_t kukaError

5.5.2.11 union { ... } kukaVal_u

La documentation associée à cette structure a été générée à partir du fichier suivant :
– kuka_api.h

5.6 Référence de la structure kukaVar_s

Définition d'une variable.

`#include <kuka_api.h>`

Attributs Publics

- char nom [KUKA_VARNAME_LEN]
- kukaVal valeur

5.6.1 Description détaillée

Définition d'une variable.

Structure définissant une variable.

5.6.2 Documentation des données imbriquées

5.6.2.1 char nom[KUKA_VARNAME_LEN]

Nom de la variable

5.6.2.2 kukaVal valeur

Valeur de la variable

La documentation associée à cette structure a été générée à partir du fichier suivant :
- kuka_api.h

Chapitre 6

Kuka API : Documentation du fichier

6.1 Référence du fichier kuka_api.h

Composants

– struct kukaAxis_s
  Description d'une configuration en fonction de chaque articulation.

– struct kukaFrame_s
  Description d'un point (position/orientation) dans l'espace.

– struct kukaPos_s
  Description sans equivoque d'un point (position/orientation).

– struct kukaError_s
  Description des erreurs.

– struct kukaVal
  Union de valeurs typées.

– struct kukaVar_s
  Définition d'une variable.

Types simples.

– typedef float kukaReal_t
– typedef int kukaInt_t
– typedef int kukaBool_t
– typedef char kukaChar_t
– typedef kukaChar_t kukaString_t [KUKA_STRING_LEN]

Types structurées.

– typedef kukaAxis_s kukaAxis_t
– typedef kukaFrame_s kukaFrame_t
– typedef kukaPos_s kukaPos_t

Enumération des types de variables.

- typedef enum kukaType_e kukaType_t
- enum kukaType_e {
  KUKA_UNKNOWN = 0x0000 | 0x000,
  KUKA_SIMPLE = 0x0000 | 0x0001,
  KUKA_INT = 0x0010 | KUKA_SIMPLE,
  KUKA_REAL = 0x0020 | KUKA_SIMPLE,
  KUKA_BOOL = 0x0030 | KUKA_SIMPLE,
  KUKA_CHAR = 0x0040 | KUKA_SIMPLE,
  KUKA_STRING = 0x0050 | KUKA_SIMPLE,
  KUKA_STRUCT = 0x0000 | 0x0002,
  KUKA_AXIS = 0x0010 | KUKA_STRUCT,
  KUKA_FRAME = 0x0020 | KUKA_STRUCT,
  KUKA_POS = 0x0030 | KUKA_STRUCT,
  KUKA_ERROR = 0x0000 | 0x000a }

Déclarations relatives aux erreurs.

- typedef enum kukaErrorType_e kukaErrorType_t
- typedef kukaError_s kukaError_t
- enum kukaErrorType_e {
  KUKA_KUKA_E = 0x0010 | KUKA_ERROR,
  KUKA_CROSS_E = 0x0020 | KUKA_ERROR,
  KUKA_RPC_E = 0x0030 | KUKA_ERROR,
  KUKA_API_E = 0x0040 | KUKA_ERROR }

Définition de la composition d'une variable.

- typedef kukaVal kukaVal

Définition d'une variable.

- typedef kukaVar_s kukaVar_t

Fonctions de base

Ces fonctions retournent 1 si elles ont généré une erreur. L'erreur est consultable via la variable statique kukaError de la lib kuka_api.

Note :

Un pointeur sur kukaError s'obtient en invoquant kuka_getError.

- int kuka_initialize (char *serveur)
- int kuka_uninitialize (void)
- int kuka_getVar (kukaVar_t *varInOut)
- int kuka_setVar (kukaVar_t *varIn)
- int kuka_loadModule (char *module)
- int kuka_unloadModule (void)

Fonctions utilitaires

- void kuka_displayVar (kukaVar_t *var)
- void kuka_getError (kukaVar_t **kukaError)

6.1.1 Description détaillée

Ce fichier décrit l'interface d'utilisation de kuka. Cette interface définit des fonctions de "bas-niveaux", et un rapport d'erreur permettant de construire par dessus une interface dont les fonctions seraient plus spécifiques (par exemple move, openGripper, ...).

6.1.2 Documentation du type

6.1.2.1 typedef float kukaReal_t

Les 'real' renvoyés par kuka seront interprétés comme des 'float' en c.

6.1.2.2 typedef int kukaInt_t

Les 'int' renvoyés par kuka seront interprétés comme des 'int' en c.

6.1.2.3 typedef int kukaBool_t

Les 'bool' renvoyés par kuka seront interprétés comme des 'int' en c.

6.1.2.4 typedef char kukaChar_t

Les 'char' simples n'existent pas en krl, il y a seulement des chaines de charactères.

Voir galement :
kukaString_t

6.1.2.5 typedef kukaChar_t kukaString_t[KUKA_STRING_LEN]

Les chaines de charactères du KRL seront interprétés comme des tableaux de char en c.

6.1.2.6 typedef struct kukaAxis_s kukaAxis_t

6.1.2.7 typedef struct kukaFrame_s kukaFrame_t

6.1.2.8 typedef struct kukaPos_s kukaPos_t

6.1.2.9 typedef enum kukaType_e kukaType_t

6.1.2.10 typedef enum kukaErrorType_e kukaErrorType_t

6.1.2.11 typedef struct kukaError_s kukaError_t

6.1.2.12 typedef struct kukaVal kukaVal

6.1.2.13 typedef struct kukaVar_s kukaVar_t

6.1.3 Documentation du type de l'énumération

6.1.3.1 enum kukaType_e

Cette enumération permet de lister les types dont nous disposons. De 10 à 19 nous avons les types simples De 20 à 29 nous avons les types structures De

Éléments énumérés :

KUKA_UNKNOWN

KUKA_SIMPLE Les types simples sont les kukaInt_t, kukaReal_t, kukaBool_t, kukaChar_t et les kukaString_t

KUKA_INT

KUKA_REAL

KUKA_BOOL

KUKA_CHAR

KUKA_STRING

KUKA_STRUCT Les types structure sont les : kukaAxis_s, kukaFrame_s, kukaPos_s

KUKA_AXIS

KUKA_FRAME

KUKA_POS

KUKA_ERROR

6.1.3.2 enum kukaErrorType_e

Enumération des différents types d'erreur.

Éléments énumérés :

KUKA_KUKA_E Ces erreur sont générées par le robot (coté serveur donc).

KUKA_CROSS_E Ces erreur sont générées par les appels de fonctions d'accés au composant com crosscommexe (coté serveur donc).

KUKA_RPC_E Ces erreur sont générées par les appels de fonctions rpc (coté client donc).

KUKA_API_E Ces erreur sont générées par les appels de fonctions de l'api (coté client donc).

6.1.4 Documentation de la fonction

6.1.4.1 int kuka_initialize (char * *serveur*)

kuka_initialize :
- se connecte au serveur rpc,
- invoque l'initialisation du cross (qui initialise COM),
- invoque la connection au cross.

Paramtres :

serveur est un pointeur vers la chaine de charactères contenant le nom du serveur ("kuka" généralement).

Renvoie :

1 si échec.
0 si succés.

6.1.4.2 int kuka_uninitialize (void)

kuka_uninitialize permet simplement de tout finaliser et de tout déconnecter.

Renvoie :

1 si échec.
0 si succés.

6.1.4.3 int kuka_getVar (kukaVar_t * *varInOut*)

kuka_getVar permet de récupérer la valeur d'une variable en fonction de son nom et de son type.

Paramtres :

varInOut est un pointeur vers la variable.

Les pré-conditions pour cette variable sont :
- le champ nom contient le nom de la variable.
- le champ valeur.type contient le type de la varable.

Les post-conditions pour cette variable sont :
- en cas de succés le champ valeur.kukaVal_u.kuka*** contient la valeur
- en cas d'echec varInOut contient une copie de kukaError.

Renvoie :

1 si échec.
0 si succés.

6.1.4.4 int kuka_setVar (kukaVar_t * *varIn*)

kuka_setVar permet de définir la valeur d'une variable en fonction de son nom et de son type.

Paramtres :

varIn est un pointeur vers la variable.

Les pré-condition pour cette variable sont :
- le champ nom contient le nom de la variable.

– le champ valeur.type contient le type de la varable.
Les Post-conditions pour cette variable sont : ∗varIn reste inchangé.

Renvoie :
1 si échec.
0 si succés.

6.1.4.5 int kuka_loadModule (char ∗ *module*)

Attention :
Ne pas utiliser pour l'instant

6.1.4.6 int kuka_unloadModule (void)

Attention :
Ne pas utiliser pour l'instant

6.1.4.7 void kuka_displayVar (kukaVar_t ∗ *var*)

Cette fonction permet d'afficher les informations d'une variable sur la sortie standard.

6.1.4.8 void kuka_getError (kukaVar_t ∗∗ *kukaError*)

Cette fonction a une double utilité :
- elle affecte au pointeur kukaError l'adresse d'une variable déclarée statique dans la lib kuka_api. Grace à ce pointeur nous pouvons consulter un "rapport d'erreur".
- elle met à jour le "rapport d'erreur".

Voir galement :
kukaError_s

Paramtres :
kukaError==NULL la fonction met à jour le rapport d'erreur.
kukaError !=NULL récupère le rapport d'erreur

Chapitre 7

Kuka API : Documentation de la page

7.1 Liste des choses à faire

Classe kukaAxis_s implémenter le type E6AXIS

Classe kukaPos_s implémenter le type E6POS

Index